UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 7,464,113 B1
APPLICATION NO. : 09/852008
DATED : December 9, 2008
INVENTOR(S) : Girkar et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On the Title Page, Item (73) in "Assignee", line 1, delete "Corporations" and insert -- Corporation --, therefor.

In column 2, line 45, after "how" delete "may" and insert -- many --, therefor.

In column 6, lines 14-15, delete "In one implementation, a shared memory counter, guarded by a concurrency latching mechanism such as a semaphore, is used to keep track of the number of outstanding transaction commits in the queue 121. In particular, the log writer process 111 checks the shared memory counter to determine if the number of outstanding transactions is not equal to the predetermined the log writer 111 to batch up the submissions to the queue 121 in terms of disk blocks, which is more efficient in terms of disk access than individually updating the queue 121 for each transaction." and insert -- In one implementation, a shared memory counter, guarded by a concurrency latching mechanism such as a semaphore, is used to keep track of the number of outstanding transaction commits in the queue 121. In particular, the log writer process 111 checks the shared memory counter to determine if the number of outstanding transactions is not equal to the predetermined threshold of transactions set by the database operator (FIG. 2, step 203). If the number of outstanding transaction commits does indeed equal the predetermined threshold of transactions, then the log writer process 111 blocks until the counter is reduced below the predetermined threshold. (As described below, this operation is performed by the net server process 123 at steps 309 and 311 of FIG. 3). On the other hand, when the number of outstanding transaction commits does not exceed the predetermined threshold of transactions, the log writer process 111 writes the transaction to one of the primary redo logs 113 (FIG. 2, step 205), submits the redo record corresponding to the transaction in the queue 121 (FIG. 2, step 207), acknowledges the commit to the primary database 110 (FIG. 2, step 209), and increments the shared memory counter (FIG. 2, step 211).

Concurrently, the net server process 123 is checking the queue 123 (FIG. 3, step 301) for redo records that have been submitted by the log writer process 111 (in FIG. 2, step 207). In one embodiment the net server process 123 checks the shared memory counter to determine if the counter is greater than zero (FIG. 3, 303). If the counter is not greater than zero, there are no redo records to be shipped to the standby system 103, so the net server process 123 loops back to step 301 of FIG. 3 where the buffer or counter is checked again. On the other hand, if the counter is greater than zero, then there are redo

Signed and Sealed this

Twenty-ninth Day of June, 2010

David J. Kappos

David J. Kappos
Director of the United States Patent and Trademark Office

records to ship to the standby database system 130. Accordingly, the net server process 123 transmits the redo record to a remote file server 131 on the standby database system 130. In response, the remote file server 131 saves the transmitted redo record in one of the standby redo logs 133 and sends an acknowledgment back to the net server process 123 that the redo record for the transaction has been written to the standby redo logs 133.

If the net server process 123 does not receive the acknowledgment (tested at step 307 of FIG. 3), then something went wrong and, accordingly, the net server process 123 takes an appropriate action, such as returning to 305 of FIG. 3 and retransmitting the redo record. On the other hand, if the net server process 123 had indeed received the acknowledgment, then the redo record is dequeued from the buffer 121 (FIG. 3, 309) and the shared memory counter is decremented (FIG. 3, 311).

Meanwhile, a primary archiver process 115 inspects the primary redo logs 113 in the background and saves the redo records in primary archive logs 117 for use in non-disaster recovery procedures. Likewise on the standby database system 130 side, a standby archiver process 135 inspects the standby redo logs 133 and saves the redo records in standby archive logs 137 and sends an acknowledgment back to the net server process 123 that the redo record for the transaction has been written to the standby redo logs 133. A managed recovery processes 139 periodically inspects the standby archive logs 137 and applies the changes to a standby database 130, which is ready to be used by the database application 100 in case of a failure in the primary database system 110.

Accordingly, an embodiment of the present invention is described in which database operators are permitted to specify how many transactions worth of data they are willing to lose in the worst case. Once the predetermined bound has been set by the database operator, no matter what happens at the primary database system 110, the standby database system 130 can lose no more than the data in the predetermined number of transactions. This guarantee, which characterizes the possible loss of data in terms of the number of transactions, is much more meaningful for database operators than a loss limitation in terms of bytes of storage or minutes of time. For example, database programmers can set a particular bound on the number of transactions that can be lost and code their database applications 100 to be intelligent about that bound when connected to the standby database system 130 in the event of a mishap on the primary database system 110. The database applications 100 can keep track of the data committed in the last number of transactions and verify which of these made it to the standby database system 130 and which did not.

ALTERNATIVE EMBODIMENTS

The present invention is not limited to the embodiment disclosed and described with respect to FIGS. 1, 2, and 3, but can be implemented in various other ways. For example, enforcement of the synchronization of a transaction commits can be performed by the primary database 110 or database application 100 before submitting a commit for the transaction to the log writer process 111, simply by checking the shared memory counter. This embodiment allows the log writer 111 to batch up the submissions to the queue 121 in terms of disk blocks, which is more efficient in terms of disk access than individually updating the queue 121 for each transaction. --, therefor.

In column 9, line 58, in claim 8, after “transactions” delete “,” and insert -- ; --, therefor.

In column 10, line 30, in claim 13, after “bound” delete “,” and insert -- ; --, therefor.